Advanced C Programming

Debugging, SAT-Tips and Efficient Algorithms

Sebastian Hack
hack@cs.uni-sb.de

Christoph Weidenbach
weidenbach@mpi-inf.mpg.de

11.11.2008

The Laws of the Edit-Compile-Debug Cycle

The Laws of the Edit-Compile-Debug Cycle

- all complex software has bugs

The Laws of the Edit-Compile-Debug Cycle

- all complex software has bugs
- the bug is probably caused by the last thing you have touched

The Laws of the Edit-Compile-Debug Cycle

- all complex software has bugs
- the bug is probably caused by the last thing you have touched
- if the bug isn't where you are lookin, it's somewhere else

Debugging Friendly Coding Style

Debugging Friendly Coding Style

- remove implicit assumptions or assert that they are valid

Debugging Friendly Coding Style

- remove implicit assumptions or assert that they are valid
- use assertions to detect impossible conditions

Debugging Friendly Coding Style

- remove implicit assumptions or assert that they are valid
- use assertions to detect impossible conditions
- don't hide bugs when you program defensively

Debugging Friendly Coding Style

- remove implicit assumptions or assert that they are valid
- use assertions to detect impossible conditions
- don’t hide bugs when you program defensively
- use a second algorithm to validate your results

Debugging Friendly Coding Style

- remove implicit assumptions or assert that they are valid
- use assertions to detect impossible conditions
- don't hide bugs when you program defensively
- use a second algorithm to validate your results
- don't wait for bugs to happen; use startup checks

Debugging Friendly Coding Style

- remove implicit assumptions or assert that they are valid
- use assertions to detect impossible conditions
- don’t hide bugs when you program defensively
- use a second algorithm to validate your results
- don’t wait for bugs to happen; use startup checks

Any Debug - Assert Code is READ-ONLY!

What Do You Do if a Bug Shows up?

What Do You Do if a Bug Shows up?

1. think

What Do You Do if a Bug Shows up?

1. think
2. run the debugger, look at the backtrace

What Do You Do if a Bug Shows up?

1. think
2. run the debugger, look at the backtrace
3. think

What Do You Do if a Bug Shows up?

1. think
2. run the debugger, look at the backtrace
3. think
4. set break points, further output, add debug code, run the debugger

What Do You Do if a Bug Shows up?

1. think
2. run the debugger, look at the backtrace
3. think
4. set break points, further output, add debug code, run the debugger
5. think

What Do You Do if a Bug Shows up?

1. think
2. run the debugger, look at the backtrace
3. think
4. set break points, further output, add debug code, run the debugger
5. think
6. remove complexity goto 1

Efficient Algorithms Through Marking

Efficient Algorithms Through Marking

1. Pointer Equality

Efficient Algorithms Through Marking

1. Pointer Equality
2. Extra Space for the Marks

Efficient Algorithms Through Marking

1. Pointer Equality
2. Extra Space for the Marks
3. Control of All Objects

Efficient Algorithms Through Marking

1. Pointer Equality
2. Extra Space for the Marks
3. Control of All Objects
4. Encapsulation including reset

Efficient SAT Implementation

Efficient SAT Implementation

1. No Search when Propagating Literals

Efficient SAT Implementation

1. No Search when Propagating Literals
2. No Search when Evaluating Clauses

Efficient SAT Implementation

1. No Search when Propagating Literals
2. No Search when Evaluating Clauses
3. Heuristic Based on Literal Occurrences